2006-03-21

Browser-Based Trust Negotiation

Cameron Morris
Brigham Young University - Provo

BROWSER-BASED TRUST NEGOTIATION

by

Cameron Craig Morris

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Cameron Craig Morris

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

_____	_____
Date	Kent E. Seamons, Chair
_____	_____
Date	Mark Joel Clement
_____	_____
Date	Yiu-Kai Dennis Ng
_____	_____
Date	Parris Egbert, Graduate Coordinator

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Cameron Craig Morris in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____ _____

Date

Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

_____

Parris K. Egbert
Graduate Coordinator

Accepted for the College

_____

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

BROWSER-BASED TRUST NEGOTIATION

Cameron Craig Morris

Department of Computer Science

Master of Science

Trust negotiation allows two parties on the Internet to establish trust in each other according to the digital credentials that each other possesses. Traditionally, trust negotiation uses certificates as digital credentials. However, certificates make trust negotiation difficult to use since people rarely have certificates available to them, and they must physically possess and secure all needed certificates in order to negotiate.

To avoid these problems, this thesis proposes that credential authorities negotiate on behalf of the user. This thesis defines Browser-Based Trust Negotiation (BBTN) as a method for negotiating with credential authorities using the Secure Assertion Markup Language (SAML).

ACKNOWLEDGMENTS

Thanks to my best friend and wife, Lisa, who has made this work possible through her encouragement, and immense sacrifice. Thanks also to my parents, Craig and Karen, for making education a priority and encouraging me along the way. Thanks to Barbara and Scott Soulier for watching the children when my studies required me to be elsewhere. Thanks to my adviser, Dr. Kent Seamons, who's patience and high standard of excellence have pushed me to accomplish more.

Thanks also to those who reviewed this thesis and gave valuable feedback: My dad - Craig Morris, my wife - Lisa Morris, Angela Bawden, Dan Jepsen, Tim van der Horst, and Greg DeHart.

This research was funded by my employer, Novell, Inc. Thanks to my managers for being flexible with me during my graduate education.

Finally, thanks to the many contributors of the various open-source projects used during this research: Apache Tomcat, Apache XML-Beans, Sun's XACML implementation, Linux, and OpenOffice.

Table of Contents

Illustration Index

Chapter 1 Introduction

There exist many potential dangers to users of the Internet when they interact with unknown web sites. Trust negotiation [2, 3, 10, 12, 19] is a recent technology that helps to avoid these dangers by establishing trust between people and unknown web sites. People must obtain and possess certificates before they can negotiate, which limits the adoption of trust negotiation. The goal of this research is to remove the certificate dependency from trust negotiation, improving the likelihood of adoption.

This research simplifies trust negotiation for users by using authoritative servers that negotiate on behalf of several people, called negotiating credential authorities. This thesis also defines a method of using negotiating credential authorities with a web browser, called Browser-Based Trust Negotiation (BBTN). BBTN simplifies the process of trust negotiation for people on the Internet, giving them improved security and privacy via trust negotiation.

This chapter introduces trust negotiation, explains the certificate dependency that impedes its adoption, presents negotiating credential authorities and BBTN as a solution, and gives example scenarios of how BBTN can be used.

1.1 Trust Negotiation

Trust negotiation helps people establish trust with unknown entities on the Internet. To negotiate, computers exchange digital credentials to learn more about the attributes of each other.

1.1.1 Digital Credentials

A *digital credential* is digital proof that one entity believes another entity to have a certain attribute. For example, suppose the Council on Higher Education Accreditation (CHEA) were to issue a digital credential that states Brigham Young University is an accredited university. BYU can use this credential to prove to other people that it is an accredited university.

Digital credentials contain attributes of an entity. Common entities are people, computers, servers, or organizations. These attributes are any property or characteristic that describe the entity, such as "this person's hair is brown," "this person's gender is female," "this organization is an accredited university," etc.

Identity credentials are credentials that uniquely identify an entity, such as "this person is George W. Bush," "this organization is Brigham Young University," or "this computer is byu.edu." Identity credentials are typically required by *closed systems,* or systems where trusted entities are known and configured in advance. Trust negotiation targets *open systems*, or systems where trusted entities are not known by name. This is done in trust negotiation by determining access according to an entity's attributes, and not their identity.

People and organizations obtain digital credentials from *credential authorities.* Credential authorities are trusted to issue valid credentials for other entities. For example, CHEA is a trusted authority of which schools are accredited universities in the United States.

1.1.2 Negotiation with Sensitive Credentials

During trust negotiation, people build trust with other entities (such as a web site) by exchanging digital credentials.

Some credentials may have sensitive information in them. With trust negotiation, users only disclose sensitive credentials when they establish a certain level of trust. An *attribute release policy* specifies the credentials another entity must disclose before an attribute is released as a credential. An attribute release policy restricts access to something unless certain conditions are met. For example, suppose a high school student creates a policy that states "Only accredited universities can access my class grades." When this student engages in trust negotiation, the policy prevents the disclosure of any grade credentials unless the other party first discloses an accredited university credential.

Figure 1 shows a typical trust negotiation protocol. When an attribute release policy is not satisfied, the negotiating service may disclose the policy so the other party learns the access control requirements. If the other party can satisfy the policy, it replies with the requisite credentials. However, these credentials can also be protected by policies when they are sensitive. The parties continue to exchange credentials and policies until either access is denied or granted.

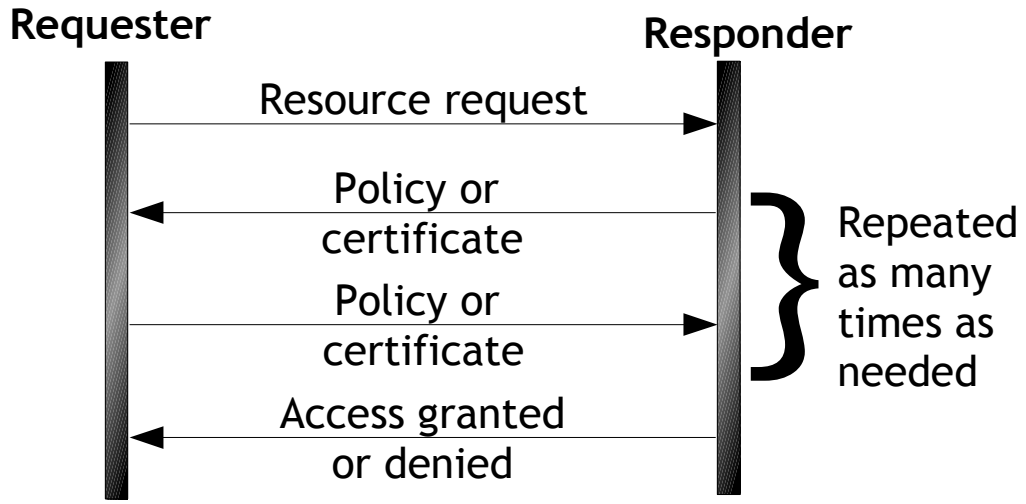

Figure 1. A Typical Trust Negotiation Protocol

1.2 Problem Statement

Currently, implementations of trust negotiation use certificates as digital credentials. Using certificates as digital credentials for users limit the adoption of trust negotiation since 1) users rarely seek out and obtain certificates, 2) issuers rarely issue certificates suitable for trust negotiation, and 3) obtaining certificates is inconvenient for users.

1.2.1 Lack of Certificates

Users seldom make the effort to obtain certificates from certificate authorities. This problem might come from the cost of purchasing certificates, the effort involved in obtaining certificates, or simply a lack of awareness that certificates exist. Secure email exemplifies this problem since it has existed for some time but few use it because of its dependence on certificates. Regardless of the reasons, the lack of certificates impedes the adoption of trust negotiation.

1.2.2 Unavailability of Adequate Certificates

Even when users seek out and obtain certificates, they are usually identity credentials and are not suitable for open systems and trust negotiation.

1.2.3 Inconvenience of Obtaining Certificates

Even if adequate certificates are available, a user will not know which attributes need to be certified until they are actually negotiating with another entity. Then the user must manually obtain the certificates from one or more certificate authorities through some mechanism such as email attachments, custom web pages, or physically obtaining the certificate via mail or in person.

1.3 Negotiating Credential Authorities

To overcome the certificate dependency for trust-negotiation, this research proposes that web servers negotiate directly with credential authorities to obtain the credentials of a user. When negotiating with a stranger, a user will refer the stranger to one or more *negotiating credential authorities*, who will negotiate on behalf of the user.

Negotiating credential authorities remove the need for users to seek out and obtain certificates. When users need credentials, their authorities provide them on-demand. However, this does assume that entities storing information about users will make this information available via negotiating credential authorities.

These negotiating credential authorities negotiate as proxies for the user. This approach means that users will not need to store and manage these credentials locally. This approach simplifies the client software since it no longer needs to negotiate. This also

provides greater mobility for the user, so long as the web server can contact the negotiating credential authorities.

These advantages simplify trust negotiation for users and increase the likelihood of adoption. BBTN is a method of using negotiating credential authorities. BBTN uses only a web browser as a client and provides a mechanism for web sites to negotiate directly with the credential authority to build trust with a user.

1.4 Examples

This section presents some examples of how negotiating credential authorities and BBTN could be used by health care providers, on-line shopping sites and government authorities.

1.4.1 Health Care

Health care involves many entities such as hospitals, insurance agencies, family practice doctors, and specialists. Each needs access to a wide array of information about a patient. These different health care parties can share patient information using BBTN.

As an example of how health care benefits from using BBTN, consider a patient coming to a family practice office for the first time. This involves filling out forms, providing insurance information, and contacting the insurance provider to discover the details of the insurance plan. Instead, insurance agencies act as negotiating credential authorities for all this information and the family practice acts as a service provider. To schedule a visit, the user simply accesses the doctor's site and introduces the insurance authority. The doctors office, if it has the proper credentials, obtains all the needed

information about the patient from the insurance authority.

In this example BBTN enables more than just information sharing. In addition, trust negotiation enables the patient to set additional policies to restrict access to sensitive information such as mental health, sexually transmitted diseases, previous abortions, etc.

1.4.2 On-line Shopping

Similarly, BBTN can use universities, colleges, and high schools as credential authorities for providing student information.

For example, suppose a student requests a discount from an on-line computer store. The student can introduce the store to the school were he or she attends. The student authenticates (or logs in) to their school web site to prove their student status to the on-line store.

Using education authorities also enable a more specific discount, tailored to the individual credentials of each user. For example, the computer store might give different discounts for part-time students, full-time students, faculty, or even current students of certain classes, such as graphic arts. Schools can establish default policies that protect students from disclosing information to untrustworthy sites.

1.4.3 Government

Negotiating credential authorities remove the need for people to physically possess digital credentials, and the same idea applies to paper credentials. Examples of paper credentials issued by the government include vehicle registration, social security number, and driver's license. Creating negotiating credential authorities in government benefits its

citizens by letting them use these credentials without physically possessing them.

For example, renting a car usually requires a driver's license. What if someone lost their wallet? As an alternative, the rental company could have the driver authenticate to his state authority and negotiate for a credential that states whether the user has a valid driver's license.

Chapter 2 Related Work

Browser-based trust negotiation uses the Secure Assertion Markup Language, or SAML [14, 15], as a foundation for digital credentials and protocols. This chapter introduces SAML assertions, compares them to certificates, and explains SAML protocols, bindings and profiles. The chapter concludes by discussing concepts found in SAML browser-based single sign-on and other related works.

2.1 Assertions and Certificates

Certificate authorities issue certificates to state that the holder has a certain identity or attribute. The certificate includes or references a public key. A person demonstrates ownership of a certificate by proving the possession of the associated private key. The verifying party may trust the issuer of the certificate directly or indirectly[1].

In SAML, an *attribute authority* (AA) issues credentials called *assertions,* that can contain authentication statements, authorization statements, and attribute statements. A SAML assertion can be of type *bearer* or *holder-of-key*. A holder-of-key assertion, similar to a certificate, contains or references a public key and requires proof of ownership of the private key. A bearer assertion, similar to a Kerberos service ticket, usually expires quickly and the bearer of the assertion can use it as a valid credential until it expires. Most scenarios use bearer assertions that expire quickly. For this reason, parties contact SAML AAs on-line to obtain assertions on-demand.

1 If the verifying party indirectly trusts an authority then it uses a certificate chain, where the chain includes certificates for subordinate authorities and a root authority.

2.2 SAML Protocols, Bindings and Profiles

SAML assertions form the base of SAML protocols, bindings and profiles. SAML protocols carry SAML assertions within protocol messages. SAML *protocols* define the structure of requests and responses. The SAML specification includes protocols to query for different types of assertions. These protocols define only high-level messages and do specify any lower-level transport protocols, such as HTTP, SOAP, or SMTP.

A SAML *binding* connects a specific transport layer to a protocol. SAML bindings [5] exist for HTTP post, HTTP redirect, and SOAP. For example, the HTTP post binding defines how to send an assertion query as a hidden attribute in an HTML form.

A SAML *profile* combines SAML assertions, protocols and bindings to solve the needs of a particular application. A profile defines the semantics of protocol messages, and can possibly restrict the syntax of assertions and protocol messages to suit the specific need of an application. Two different implementations of the same profile must inter-operate.

2.3 Single Sign-On Profiles

Most common profiles of SAML are for browser-based single sign-on applications. All browser-based profiles use bearer assertions to achieve single sign-on (i.e., one authentication across different domains) with a zero footprint client (i.e., no additional software installed beyond the browser).

2.3.1 SAML version 1 SSO

The SAML version 1 specifications [14] includes two profiles for browser-based single sign-on.

In both profiles, a user authenticates to an AA (a typical web-page portal) that can redirect the user's browser to another web site, called a *service provider* (SP), without having to re-authenticate. The AA sends a bearer assertion, containing an authentication statement, to the SP via the browser. The SP uses this *authentication assertion* as proof that a user has authenticated to the AA[2]. After an SP receives and verifies the authentication assertion, the SP grants the user appropriate access.

Using either profile, an AA can provide single sign-on to an SP when users click a special link on its web page. This special link will send the browser to the *intersite-transfer service* on the AA that will prepare an assertion specifically for the SP, and then send the browser to the SP.

The two profiles differ in the redirect method used to send the browser from the AA to the SP. The Browser-Post profile sends the assertion to the SP by placing the assertion in an HTML form. When the form loads in the browser, javascript submits the form to the SP using the HTTP post method.

The second profile, called Browser-Artifact, sends a simple HTTP redirect to the browser with a destination URL. Placing an assertion within a URL may exceed the URL length limit, so an artifact that represents the assertion is appended to the URL query string. The artifact contains enough information for the SP to identify and query the AA for the assertion.

The SP queries the AA for the assertion through a direct channel, also known as the

2 Alternatively an attribute authority sends an artifact of the assertion, in which case the service provider will contact the attribute authority directly to resolve the artifact into an assertion.

back channel. The SP can also query the AA for additional *attribute assertions* containing attribute statements, such as gender, shipping address, etc. Figure 2 shows the artifact profile for browser-based single sign-on (for SAML version 1).

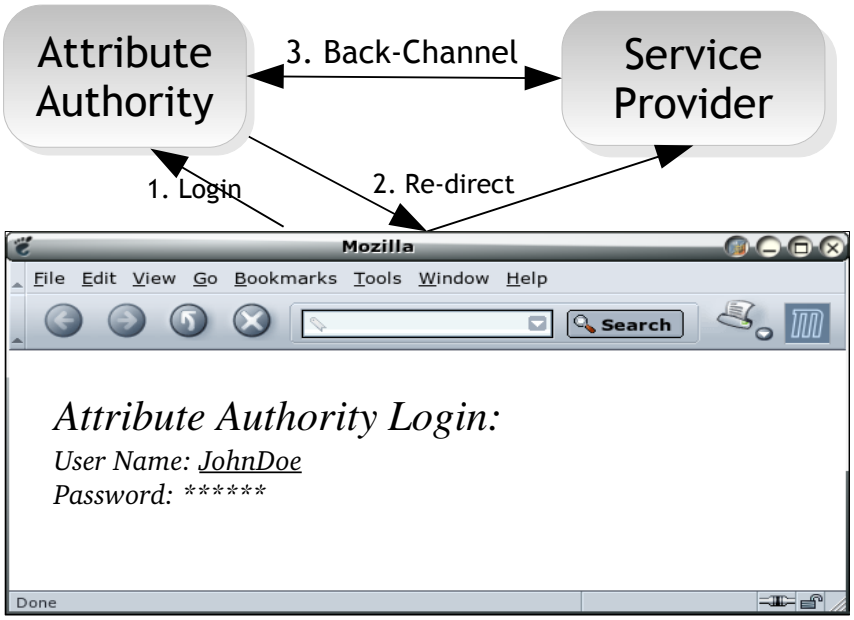

Figure 2. Browser-Based Single Sign-On

Two prominent technologies use and extend these profiles: Liberty Alliance [11], and Shibboleth [4]. The SAML profiles provide single sign-on for users that first authenticate to an AA and then wish to go to an SP. However, this is not sufficient for users that first contact the SP and may not know which AA will authenticate the user. Liberty Alliance and Shibboleth both defined new profiles that allow an SP to send an *authentication request* in a redirect to the AA.

2.3.2 Shibboleth

Shibboleth defines a separate service known as the "Where are you from?", or WAYF, for unauthenticated users that come to an SP. The WAYF allows a user to identify

an appropriate AA. The WAYF service generates an authentication request and sends it to the AA by redirecting the browser to the AA. The AA authenticates the user and sends an anonymous authentication assertion in a browser redirect to the SP. The SP uses this information to query the AA on the back channel for the attribute assertions pertaining to the user. Access to resources on the SP is determined by the attributes that the AA provides.

2.3.3 Liberty Alliance

Like Shibboleth, a Liberty Alliance SP can send an authentication request to an AA using a browser redirect. However, Liberty Alliance did not define a separate service for this.

Liberty Alliance defines a number of useful extensions to SAML version 1. Identity Federation links accounts from an AA and SP and uses a pseudonym for these accounts when providing single sign-on. Liberty Alliance also defined *metadata*, documents that describe an AA or SP. (Note that in Liberty Alliance, the authenticating server is called an identity provider, and not an AA.) Instead of using attribute queries on the back-channel, Liberty Alliance defined an extensive web services framework.

2.3.4 Single Sign-On with SAML v2

Most of the concepts used in Liberty Alliance were added to the SAML 2.0 specifications, with the exception of the web services framework. Figure 3 demonstrates the use of an authentication request (AuthnRequest) in the web single sign-on profile as defined in SAML version 2.0.

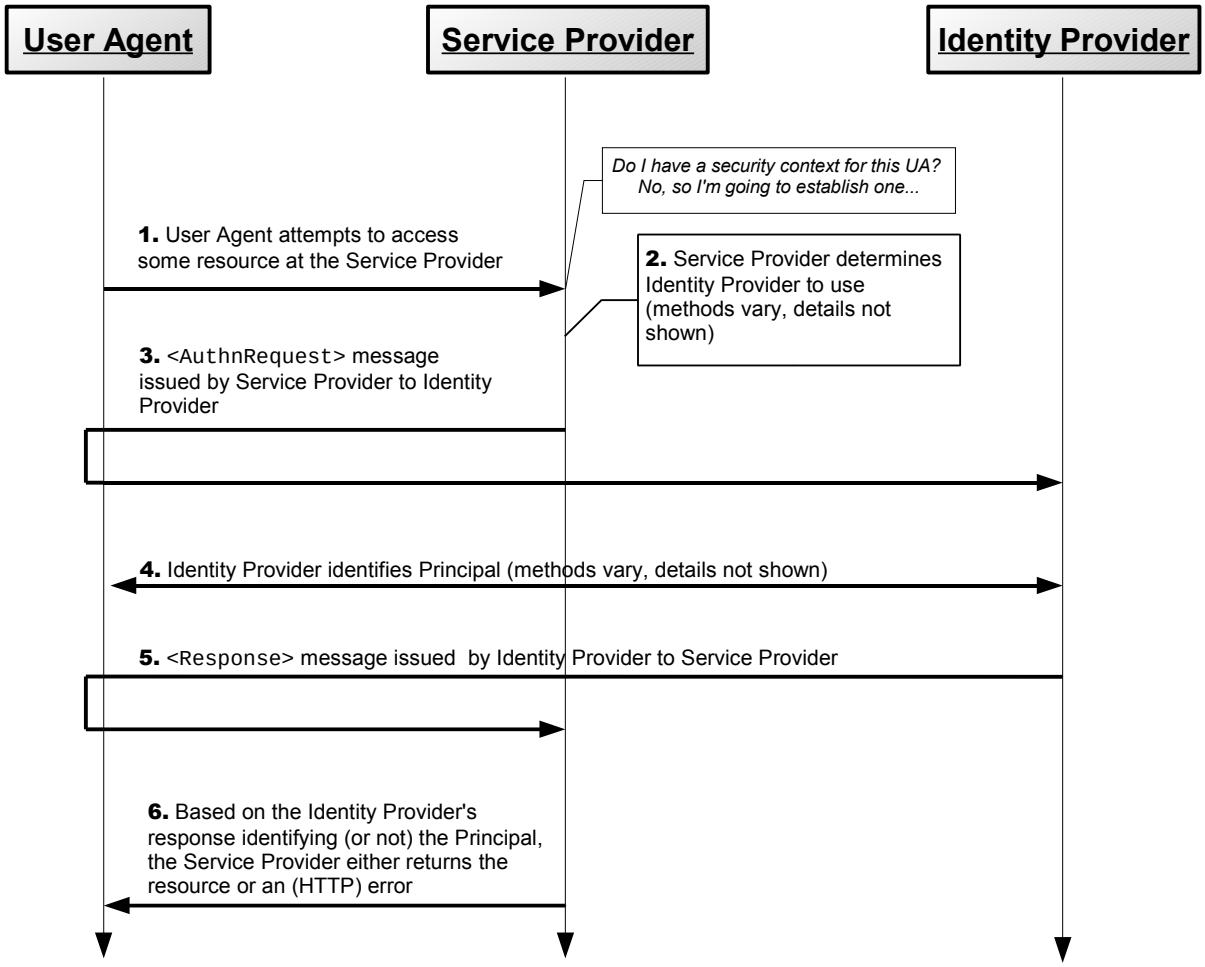

Figure 3. Web Single Sign-On Profile showing an AuthnRequest
(Reproduction of a diagram found in the OASIS specifications [15])

The SAML version 2 specifications also adopted the Liberty Alliance naming conventions. Note that an AA and an identity provider usually refer to the same entity. This thesis refers to the identity provider as an AA because the focus is on attribute based systems instead of identity based systems.

2.4 Real-Time Release

Pfitzmann and Waidner [17] raise a concern about the lack of "real-time release" of attributes in single sign-on proposals, specifically in Shibboleth and Liberty Alliance version 1. They define real-time release as the ability to prompt the user for consent before releasing attributes. Their Browser-Based Attribute Exchange (BBAE) [16] profile addresses this concern.

In BBAE, Pfitzmann and Waidner suggest an AA should disclose all attributes *before* instead of after redirecting back to the SP. This means the web browser will be interacting with the AA instead of the SP. This allows the AA to prompt the user for consent to release an attribute.

In response to BBAE, Liberty 1.2 addressed this problem by defining three different methods of prompting the user:

1. Trust the SP to prompt the user,
2. Redirect back to the AA to prompt the user, or
3. Prompt the user through an external mechanism, such as a phone or an instant messenger service.

Shibboleth, Liberty Alliance and BBAE each incorporate SAML to create single sign-on technologies. Browser-Based Trust Negotiation also builds on SAML as a foundation, and borrows many of the concepts found in these other technologies.

Chapter 3 Browser-Based Trust Negotiation

Browser-based trust negotiation (BBTN) combines SAML single sign-on technologies with trust negotiation to create negotiating credential authorities. This chapter presents an overview and message flow of BBTN, defines the SAML trust negotiation protocol (SAML TN), explains real-time release of attributes in BBTN, and describes the visual policy resolver (VIPR).

3.1 Overview of BBTN

BBTN relies on a standard web browser and does not require specialized client negotiation software. When a user requests a protected resource from an unfamiliar web site (SP), BBTN helps to build trust with the SP web site. One or more AAs negotiate with the SP in behalf of the user to establish this trust. Figure 4 shows how the different parties involved in BBTN interact generally.

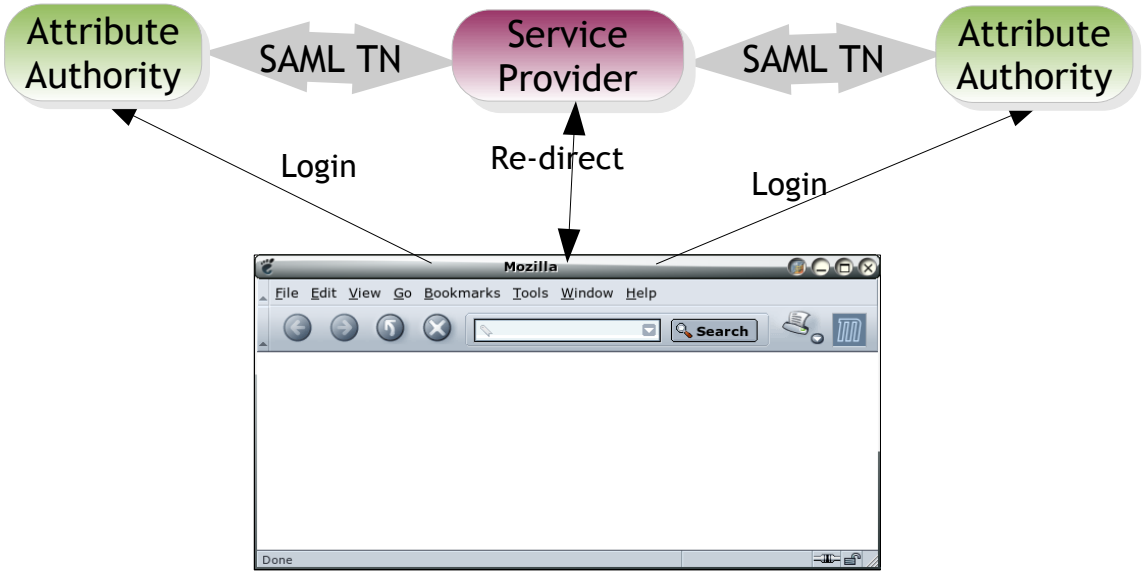

Figure 4. Different Parties Involved in BBTN

The user informs an unknown SP which AAs will negotiate for him. The SP requests an authentication assertion from an AA with an authentication request. Afterwards, the AA negotiates as a proxy for the user with the SP on the back-channel. The back-channel negotiation uses the SAML TN protocol defined in section 3.3.

In BBTN, an AA's trust in an SP is an open system because this trust is not identity based. However an SP's trust in an AA is still identity based. The trust that an SP places in a person is attribute based and is considered an open system.

3.2 BBTN Message Flow

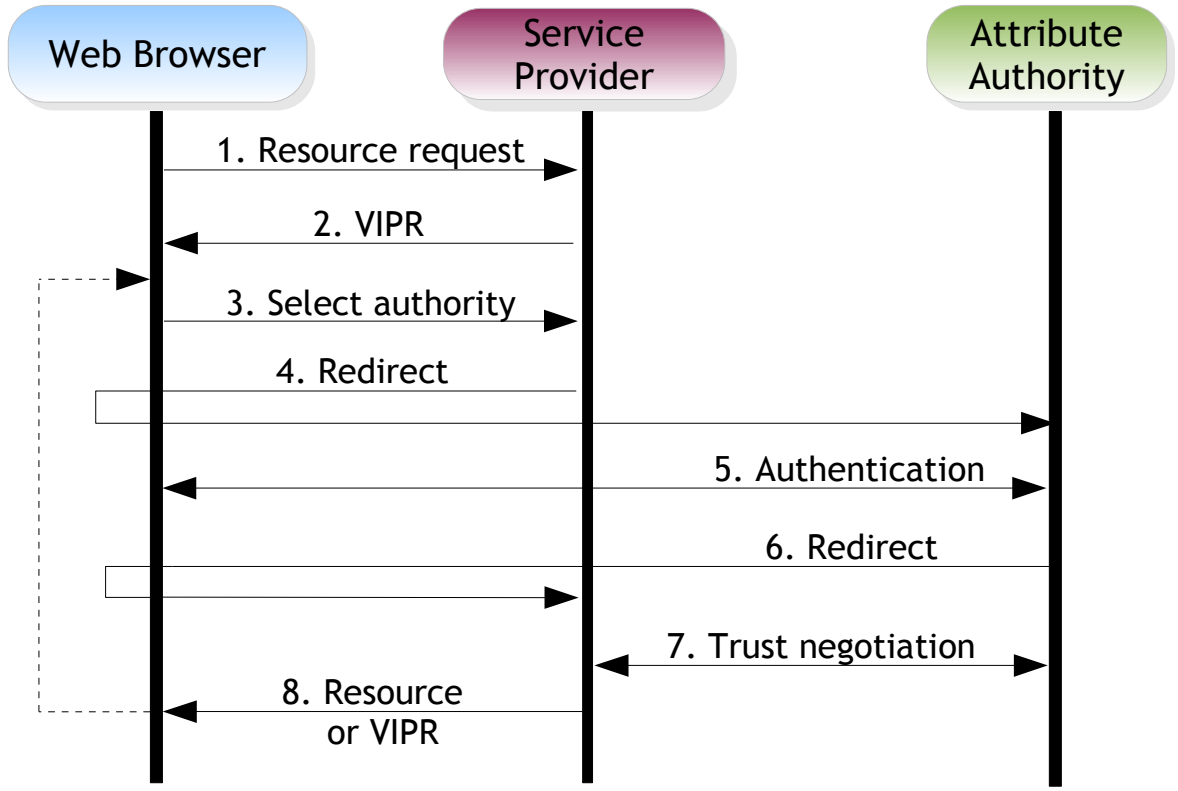

Figure 5. BBTN Message Flow

Figure 5 shows the message flow of BBTN. The following steps to this procedure combine trust negotiation with SAML so that users can establish trust with web sites using

a common web browser and negotiating credential authorities:

1. A user requests a protected resource at an SP.
2. The SP returns a visual policy resolver (VIPR) page to present the policy that protects the resource.
3. The user selects a suitable AA for all or part of the policy.
4. The SP redirects the user to the selected AA. The SP includes an authentication request in the redirect.
5. The AA authenticates the user.
6. The AA redirects the browser back to the SP and includes an authentication assertion.
7. The SP and AA negotiate trust using SAML TN on the back channel.
8. The SP either releases the resource or presents another VIPR page, in which case, steps 3-8 repeat until the user gains access to the resource.

3.3 Trust Negotiation Protocol for SAML

This section extends the existing SAML protocols to create a negotiation protocol, called SAML TN. A schema definition for this protocol, suitable for SAML version 2, appears in the Appendix at the end of this paper.

All protocols in SAML follow a synchronous request/response model. All responses in SAML use a common response document can contain assertions and always contains a status. The status includes three elements, 1) StatusCode, indicating either success or an error, 2) StatusMessage, an optional text message describing the status of the request, and 3) StatusDetail, an optional field that can contain any information relevant to the status of

the request. In SAML TN, the StatusDetail contains the policy controlling access to the requested request.

To support trust negotiation within the existing request/response model, a new request must be defined and is known as a NegotiationMessage. A NegotiationMessage includes a response document that can contain either assertions or policies. Figure 6 shows how the NegotiationMessage and SAML Response can be used in conjunction with any SAML protocol to implement trust negotiation.

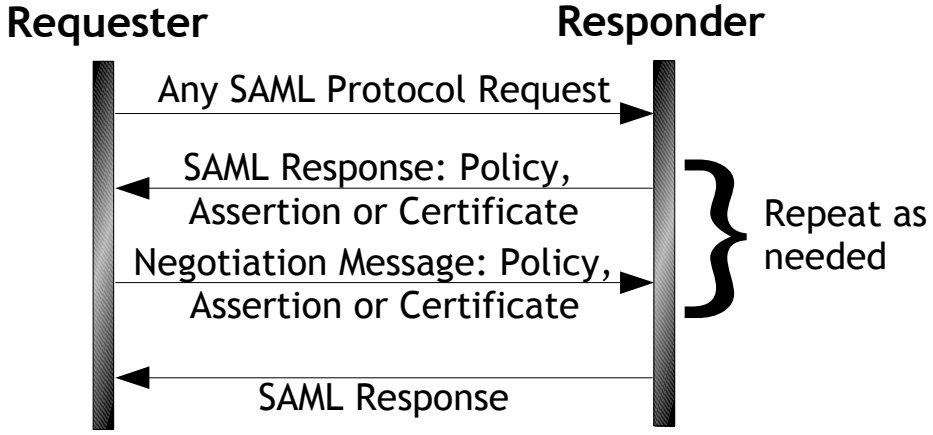

Figure 6. SAML Trust Negotiation Protocol

SAML TN can use short-term bearer assertions and holder-of-key assertions as credentials. However, AAs will most likely use bearer assertions and SPs will most likely negotiate with holder-of-key assertions. This lets SPs hold the assertions for a much longer period of time than a bearer token. SPs that handle a high volume of requests cannot afford to dynamically obtain bearer assertions for each negotiation.

The SP must prove to the AA that the SP knows the private key associated with holder-of-key assertions. One way to do this is for all the SP's assertions to reference a

single public key that the SP uses for session authentication with the AA. This could be done using mutual authentication of a TLS session or by the SP signing all messages sent to the AA.

3.4 Real-Time Release in BBTN

This section explores the real-time release options from BBAE and Liberty to determine which will work best for BBTN

In BBAE, the AA initiates the attribute exchange with the SP and redirects the user to the SP after the requested attributes are released. The policies at AAs and the SP may create inter-dependencies across several AAs. These dependencies cause one AA to pause its negotiation until an SP completes negotiation with another AA. If an AA waits to complete negotiation before returning the browser to the SP, then either deadlock occurs or the negotiation would have to be abandoned.

This problem could be avoided by creating additional redirections. To accomplish this, one SP would redirect the browser to the SP and then on to another AA for attribute release and then back again. This would make the protocol less efficient and more complex. Similarly, using the Liberty Alliance method of redirecting back to the AA for attribute release creates redundant redirections. To avoid these inefficiencies, BBTN does not return to an AA after the authentication assertion is sent to the SP.

Unlike Liberty Alliance, the AA in BBTN is not required to have any previous relationship or trust in the SP. Therefore, BBTN cannot rely on the Liberty Alliance method of trusting the SP to prompt the user for permission to release an attribute.

Instead, BBTN could use the third method defined by Liberty, an external attribute release mechanism. Since instant messenger services have become almost as common as web browsers, an instant messenger client serves well as the external mechanism to prompt the user for attribute release.

3.5 Visual Policy Resolution Service

The SP for BBTN needs a service to help the user resolve access policies by authenticating to one or more AAs. Although similar to the Shibboleth WAYF, and other authentication request services, this service must create a web page to visually explain the access control policy of a requested web page to the user. This service interacts with the user, allowing the user to select different AAs, and provides feedback by showing the progress of negotiation. For BBTN we call this service the Visual Policy Resolver (VIPR).

The VIPR allows the user to identify multiple AAs, one at a time. Once the user selects one, the VIPR sends an authentication request to the selected AA. After authentication via the AA and trust negotiation on the back channel, if the user still cannot access the resource, the SP again presents the VIPR page, giving the user an opportunity to choose another AA.

Every time the VIPR page is presented it shows the progress of the negotiation by displaying satisfied and unsatisfied parts of the policy. When a request for a protected resource fails, the VIPR does not return a generic error message, such as “Error code 505: access denied”. Instead the VIPR provides information as to why a request fails, such as “Access to this resource requires the credentials a, b and c from any of the following

authorities: x, y, or z".

Chapter 4 Implementation

This chapter describes a prototype implementation of BBTN. It details the sub-components of the implementation: the SAML negotiator engine, the AA, the SP, and the VIPR. It also discusses challenges and pitfalls encountered during implementation.

The implementation of BBTN required several components on both the SP and the AA. Figure 7 shows an implementation architecture diagram of the AA and SP.

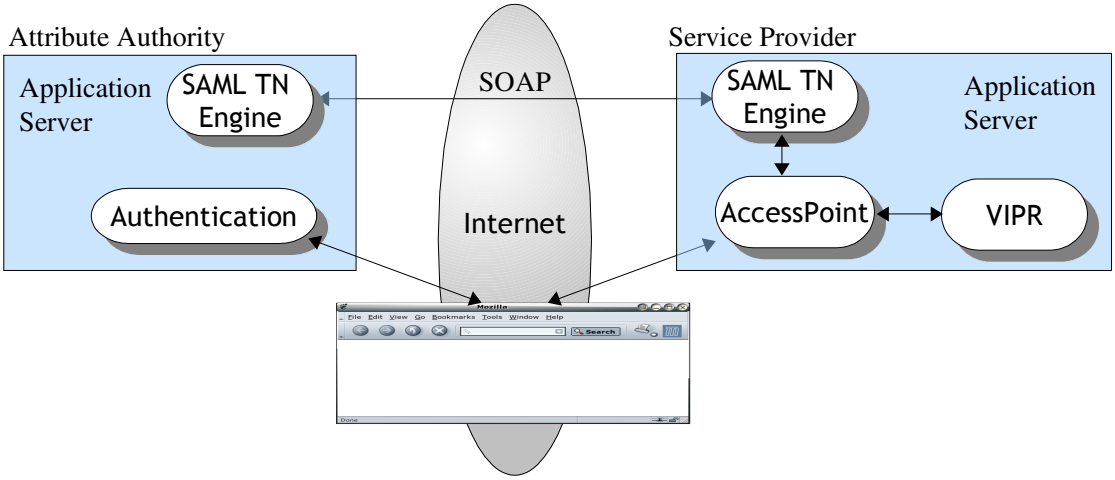

Figure 7. BBTN Implementation Architecture

4.1 SAML Trust Negotiation Engine

Both the AA and the SP use a SAML trust negotiation engine to handle SAML protocols and policy decisions. Figure 8 shows the architecture of the SAML trust negotiation engine. The PDP from the open source XACML policy project (SunXACML [13] by Sun Microsystems) was adapted to return a policy when a request fails.

The Policy Enforcement Point (PEP) calls the PDP and enforces the decision returned. It derives, from an incoming policy, which attributes another party requests and releases them if possible.

The SAML Negotiator uses the SAML version 2.0 specification for assertions and protocols. The SAML Negotiator handles the SAML requests and maintains the open negotiation sessions.

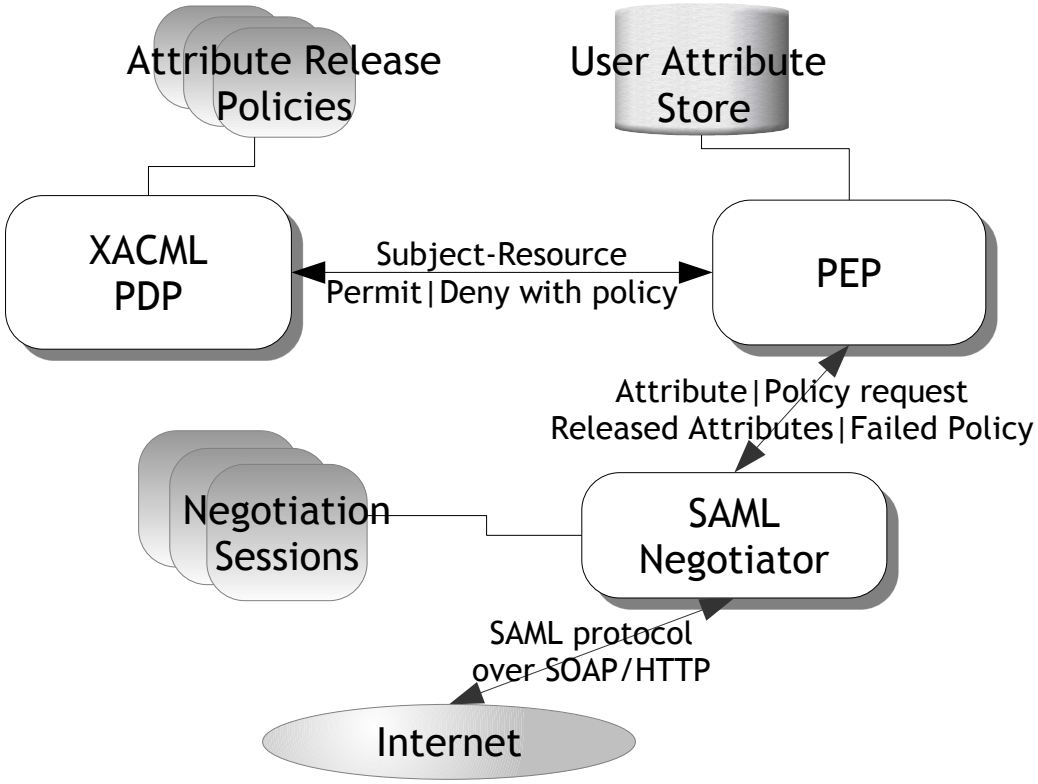

Figure 8. SAML Trust Negotiation Engine Architecture

The negotiation messages are exchanged using the SAML SOAP binding as defined in the SAML version 2 binding specification. The SOAP binding limits the exchange to a synchronous request-response, so the SAML negotiation engine must keep the state of negotiation across many requests and responses. It does this by tracking the 'ID' attribute

of requests and 'InResponseTo' attribute of responses and matching them to the appropriate negotiation session.

4.2 Attribute Authority

The attribute authority (AA) uses two web servlets, an authentication servlet, and a SOAP back-channel servlet that uses the SAML Negotiator for attribute requests and negotiation requests.

The authentication servlet implements the web single sign-on profile of SAML version 2.0 (using only the post binding). It accepts post messages that include authentication requests. The servlet authenticates the user with a simple user-name and password, and then redirects the browser back to the SP including an authentication assertion.

4.3 Service Provider

The SP uses two servlets, the *AccessPoint* and the *AuthenticationRequester*. All resource requests channel through the AccessPoint servlet. This servlet uses, as subcomponents, the SAML Negotiator engine and the VIPR service. When the engine denies access to a resource, the AccessPoint uses the VIPR service to generate a VIPR web page containing links and buttons for contacting AAs.

The links and/or buttons in the web page generated by the VIPR posts the selected attribute authority to the AuthenticationRequester servlet. This servlet locates the appropriate AA endpoints and sends an authentication request document. The request

indicates where the AA should redirect the web browser after the user authenticates.

The user's browser then returns to the AccessPoint servlet (at the SP) with an assertion (from the AA). The AccessPoint uses the SAML Negotiator engine to negotiate on the back channel with the AA. If the policy still requires more information, the SP once again generates a VIPR page.

4.4 Visual Policy Resolver

When access to some page or service fails, the VIPR service generates a web page that allows the user to introduce an AA in order to satisfy the policy that failed. In this implementation, the VIPR service parses the failed policy into a simplified tree. The tree has 'AND' and 'OR' branches with attribute leaf nodes that specify the required attribute and required issuer. The VIPR uses this tree to create an HTML tree that lets the user graphically see the logic required to satisfy the policy.

The leaf nodes in the tree provide links, buttons or text fields that allow the SP to contact a specified AA. When parsing an XACML policy, the VIPR generator creates leaf nodes in the tree for each 'Subject attribute designator' element. These elements specify which credentials a subject must have, and who must issue the credentials.

The policies cannot express indirect trust without an extension to the XACML language. So a policy can only specify root authorities as issuers of credentials. The implementation requires separate configuration for subordinate authorities.

The implementation allows three possible leaf nodes:

1. **No issuer:** Any AA, that can give the requested attribute, will satisfy the leaf

node of the policy. The VIPR page presents this case as an empty text input box that lets the user type which authority to use.

2. **Direct issuer**: A policy specifies an authority, and no configuration exists for subordinate authorities. For this type of the leaf node, the VIPR presents a simple submit button.
3. **Root Authority issuer**: A policy specifies an authority, and configuration exists to indicate support for subordinate authorities. The VIPR presents this node as an empty text input box for the user to type in a subordinate authority. A selectable pull-down list appears in the HTML node if the SP already knows of some acceptable subordinate authorities.

Figure 9 shows an XACML policy and Figure 10 shows a VIPR web page generated from this policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyID=”SongDiscountPolicy"
...
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-equal">
   <AttributeValue      DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag-size">
      <SubjectAttributeDesignator
          AttributeId="DriversLicense"
          DataType="http://www.w3.org/2001/XMLSchema#string"
          Issuer="dmv.gov"/>
     </Apply>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">enrolled</AttributeValue>
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator
             AttributeId="StudentStatus"
             DataType="http://www.w3.org/2001/XMLSchema#string"
             Issuer="dedu.gov"/>
      </Apply>
   </Apply>
</Condition>
</Rule>
</Policy>
```

Figure 9. Sample XACML policy (used to generate Figure 10)

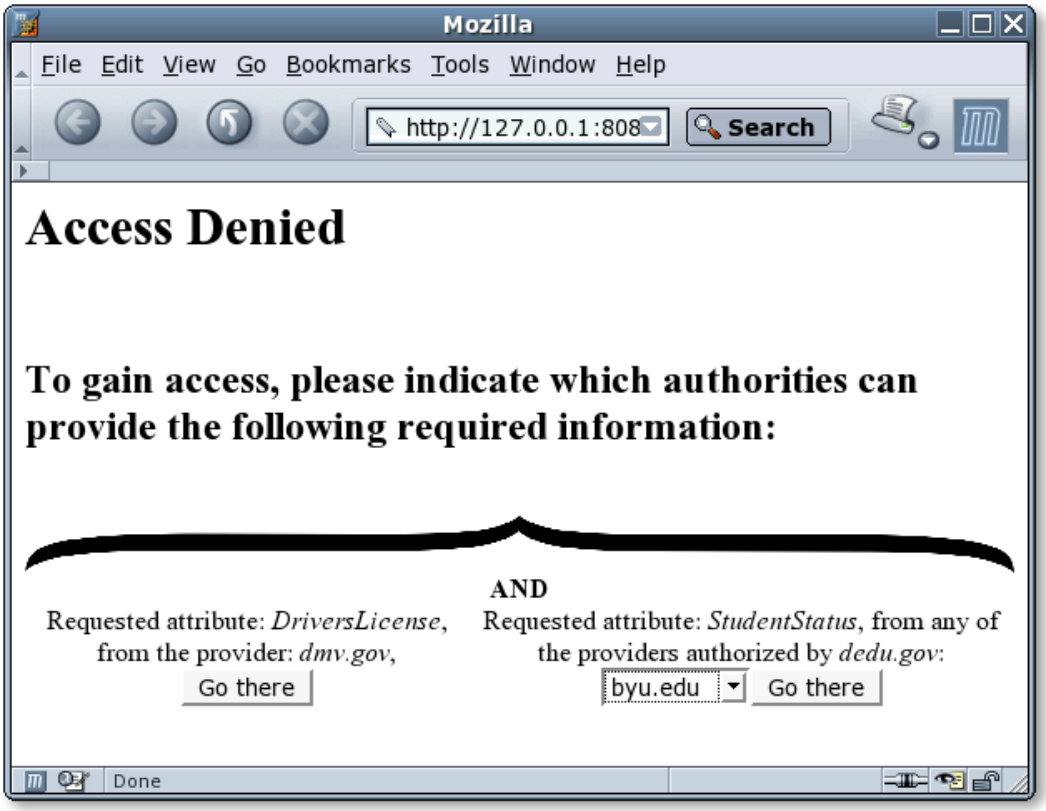

Figure 10. Generated VIPR Web Page

4.5 Challenges and Pitfalls

Even thought BBTN removes the need for client software, it complicates the implementation of the AA and SP. The main challenges encountered during implementation include selecting the correct AA to contact during negotiation and detecting request loops.

4.5.1 Selecting the Correct Attribute Authority

A challenge specific to the SP is selecting the correct AA with which to negotiate. The SP starts a negotiation session with each AA authority as the user introduces them. This works fine if the negotiation with each AA completes independently. However, a negotiation with one authority is suspended if it depends on completing a negotiation with a different authority. When one negotiation completes, and multiple negotiations sessions are still suspended, which session should resume next? An incorrect choice may result in redundant message exchanges.

The implementation contacts AAs in the reverse order in which the user selects them. For each message exchange, policy requests are either pushed on, or popped off the request stack. The stack also keeps track of which authority made the request. As negotiations with different authorities complete, previous requests are removed from the stack and an SP contacts the associated authority to resolve the request.

4.5.2 Detecting Request Loops

Another challenge is detecting loops and knowing when to terminate negotiation. If attribute release policies across several parties have a circular dependency, an infinite loop

occurs during negotiation. For example (see Figure 11), suppose a user requests a web page from an SP. To release the web page, the SP requires credential *a1* from AA_1. To release credential *a1,* AA_1 requires credential c1 from the SP. To release credential c1, the SP requires credential a2 from AA_2. To release credential a2, AA_2 requires *a1* from the SP. If the cycle is not detected, the SP would then request *a1* from AA_1 and repeat the process.

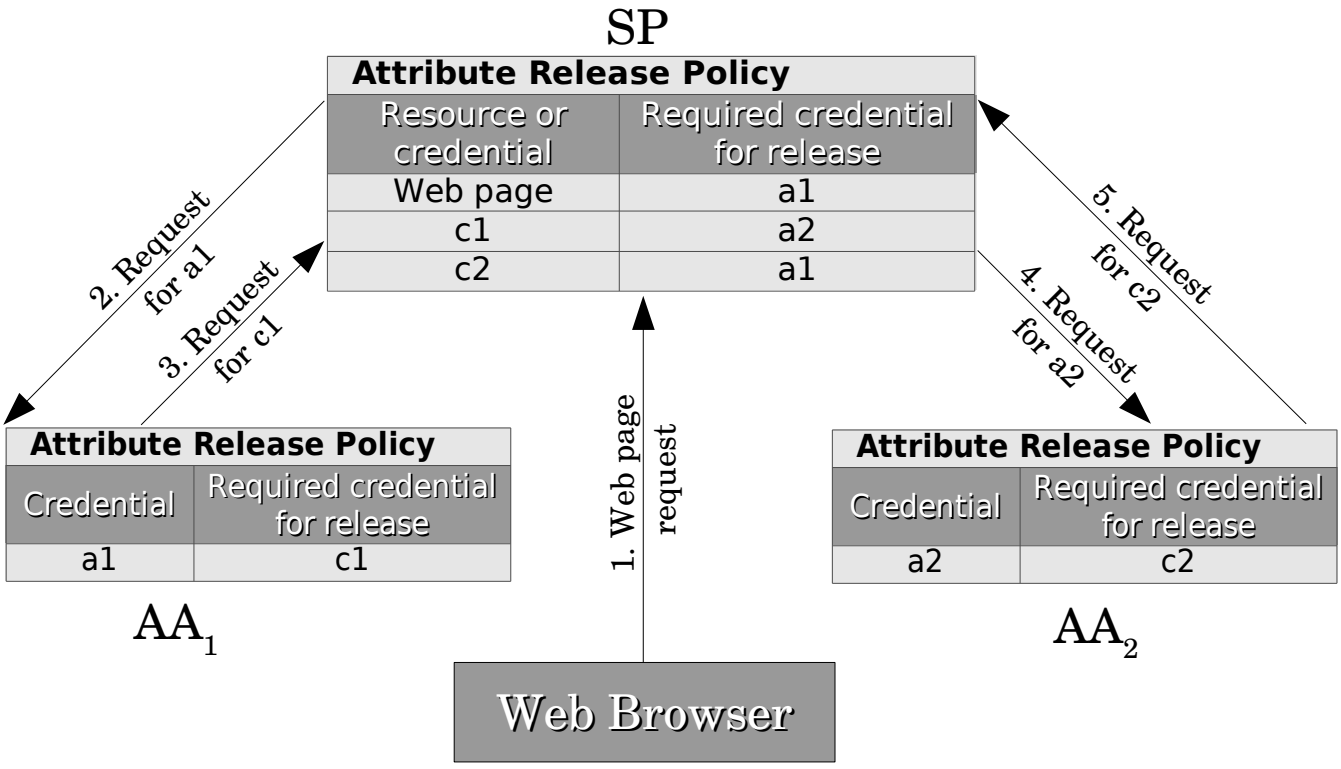

Figure 11. Dependency Loop Example

The SP implementation detects loops by saving a set of outstanding credentials requested. When a request fails, negotiation continues only if the policy that failed requests some additional credentials not found in the set of outstanding credentials requested. This prevents loops and redundant message exchanges.

4.6 Implementation Overview

Items left for future work include signing and encrypting SAML messages and real-time release of attributes via an instant messenger. Overall, the prototype implementation successfully achieved the following:

1. Protects web resources and credentials at the SP via XACML policies
2. Redirects users from an SP to an AA with an authentication request
3. Authenticates the user at the AA and sends an assertion to the SP
4. Protects user credentials using XACML policies and trust negotiation

Chapter 5 Analysis of BBTN

This research removes the dependency of trust negotiation on certificates issued to users, but at what cost? How does this approach effect privacy and security compared to current single sign-on proposals and current trust negotiation proposals? The following sections answer these questions by analyzing BBTN using the following criteria: credential management, policy management, usability, performance, privacy, unintended credential usage, denial of service attacks, and other security threats.

5.1 Credential Management

The addition of SAML assertions as a credential to trust negotiation simplifies credential management for both administrators and users.

In BBTN, the user does not have to seek out and obtain certificates from various organizations. This eliminates the need to adhere to stringent rules, as stated in a certificate practice statement (CPS). These rules may require the user to request certificates in person, present certain forms of identification, or store signing keys in specially protected environments.

Furthermore, each certificate authority may have different mechanisms of sending the certificates to the user, such as by disk, mail, email, or through a custom built web page. After receiving the certificates and private keys, the user must secure them to protect from theft.

BBTN also removes the need for administrators to issue, revoke, and distribute certificates to users. Such activities could happen very frequently if a user's attribute

changes frequently, such as a role.

The following university scenario, adapted from an example in Shibboleth [4], demonstrates this burden. Students and professors from different universities wish to collaborate on their research. Students of one class need documents hosted on a different university's protected web page. In order to accommodate this collaboration, students need to have certificates created at the beginning of the semester. Administrators need to make the certificates expire at the end of the semester and also revoke certificates when classes are dropped.

Using BBTN simplifies the model greatly. Rather than issue certificates every semester, administrators create user accounts only once. Since current class enrollment determines the access to other universities, administrators have nothing further to configure.

5.2 Policy Management

Moving credentials off of the user's computer onto the AA also moves the policy enforcement point to the AA. Before, credentials from many different authorities relied on the user to secure access to these credentials. In BBTN, that responsibility resides on the AA. The authority that issues a credential must also secure access to the credential.

Both organizations and users will most likely want the AA to create and enforce default attribute release policies. Many organizations worry about the privacy of the user's data they store, especially when legislation imposes severe penalties for privacy violations. These organizations would probably want to enforce a default policy for all their users'

data. Also, creating attribute release policies requires specialized skills, and many users will not craft adequate policies without additional training. Since the enforcement point resides on the authority, the authority can apply a set of default policies for every user.

Some users will also wish to create their own policies. BBTN satisfies this desire by allowing user-defined policies in addition to the default policy.

Other single sign-on systems also have attribute release policies. Shibboleth and implementations of Liberty Alliance use identity-based attribute release policies, not attribute-based as in trust negotiation. This means that the user assigns rights to credentials for each SP one by one. Attribute-based release policies, once in place, apply to any SP. This allows the user to exchange credentials securely with any number of SPs.

If single sign-on technologies continue to gain momentum in the industry, many of the web sites people visit will be SAML enabled SPs. With current single sign-on proposals, a person can very easily forget and mismanage which web sites he has granted access to his attributes. In this scenario, trust negotiation has a compelling advantage since creating one release policy applies to all possible SPs.

5.3 Usability

A browser-based application for trust negotiation simplifies the usability for a user. First, the user does not have to install or upgrade trust negotiation software. Second, most computer users feel comfortable using a web browser. Third, using negotiating credential authorities moves credential configuration and management duties to the AA, relieving the user from these duties.

A web-based application also has drawbacks. HTML does not provide the rich flexibility that a native user interface provides. Complex tasks are more difficult to implement and can be more cumbersome to use when implemented as a web page than if implemented as a native client.

5.4 Performance

How does BBTN compare to traditional trust negotiation (TN) in terms of efficiency and scalability? This section compares the number of message exchanges and user interactions between the two approaches.

5.4.1 Message Efficiency

For a given negotiation involving certain policies and credentials, the number of messages in BBTN will exceed the number in TN whenever BBTN distributes the negotiation across several AAs. For instance, in a negotiation between a server and either a client or single AA, the server can send the full policy in a single message. The negotiation in TN and BBTN will be the same. The server assumes all credentials relevant to the user are possessed by the client or AA. If instead the user's credentials are distributed among several AAs, the server will disclose the policy, or relevant portions, to each AA. Thus, an upper bound on the number of messages using BBTN is the number of messages using TN multiplied by the number of AAs, plus the messages for redirection and authentication that occur once for each attribute authority. The number of BBTN messages compared to the number of TN messages is represented by the following formula:

$$TN + c \cdot n \leq BBTN \leq TN \cdot n + c \cdot n$$

n = number of attribute authorities
c = constant number of messages used for redirection and authentication
TN = number of messages needed for traditional trust negotiation
$BBTN$ = number of messages needed for BBTN

This formula indicates that the number of messages in BBTN at most grows linearly as the number of AAs increases.

5.4.2 User Prompt Efficiency

Each AA prompts the user for authentication once per session. Negotiations involving more than a few AAs could annoy users with an excessive number of authentication requests. The technology exists in Liberty Alliance [11] and SAML 2.0 [15] to achieve single sign-on, not only from an AA to an SP, but also to other AAs. To accomplish this each AA also acts as an SP and users can federate their identity across the domains, providing one authentication across all domains. Using this technique for BBTN can limit the number of authentication prompts to one.

5.5 Privacy

5.5.1 Credential Linking

Credential linking is a privacy threat where SPs learn more about the user than what the user disclosed by linking credentials or information that a user discloses to several SPs. For example, suppose that during trust negotiation, an SP for a grocery store learns the gender of a visitor. Suppose that this grocery store also has connections with an on-line magazine shop. If the gender credential contains linkable information, the grocery

store can learn if the user had visited the magazine shop as well as any information the user disclosed to the magazine shop. For example, the grocery store could learn about the interests of the person from the articles viewed at the magazine shop and then present advertisements specific to his interests.

If the credential contains a uniquely identifying attribute, such as a social security number or a drivers license, then nothing can prevent linking. But, if the credential contains something generic such as hair color, citizenship, or gender, then the credential format should not allow linking.

Certificates and holder-of-key assertions can be linked using the public key embedded in them, even the attribute in the credential contains generic information. This allows an SP to store information about the user, such as which pages he visits, and which credentials he discloses. Once a user discloses the same credentials to other SPs, the SPs can then aggregate all the data and credentials, creating access to information that the user never intended.

Assertions can contain a name identifier. Shibboleth and Liberty Alliance do not produce assertions that contain a linkable name identifier. Liberty Alliance and SAML 2.0 define a *persistent* name identifier (or pseudonym) that remains constant for a particular SP. However, the identifier still remains unique across different SPs, and SPs cannot link them. These specifications also define *transient* (or one-time use) name identifiers which are small random numbers used only to identify the user's current session.

BBTN prevents linking of bearer assertions since it only uses transient or persistent

identifiers. Using these identifiers, nothing else within SAML assertions or protocols (other than the attributes themselves) can link user's credentials. This adds some level of protection against linking. There exist additional methods to link users, such as tracking which Internet addresses the user comes from. These threats are not discussed here since they exist external to SAML and BBTN, and other proposals exist to deal with them.

5.5.2 Tracking People's Browsing Habits

While SAML assertions help to prevent linking, BBTN, and SAML in general, have an additional privacy concern, the ability to track people's browsing habits. Because an AA negotiates on behalf of the user, it can store information about which sites the user visits. Pfitzmann and Waidner [16] identified this concern with other single sign-on proposals. As a way to minimize this they suggest that SPs use one common URL and an 'opaque handle' when it requests an authentication. The opaque handle identifies the user's current session and is meanful only to the SP.

This suggestion still allows AAs to track to which SPs a user browses, but minimizes the amount of information the AA can glean. For example suppose a user requests single sign-on to a hospital. An AA's log for a user may look something like "http://somehospital.com/drug-abuse-clinic, http://somehospital.com/abortion-clinic". Using the suggestion from BBAE, the same log could look like "http://somehospital.com?ID=12345, http://somehospital.com?ID=23456", giving no indication as to what pages were viewed at the hospital.

A more effective alternative makes use of the enhanced client/proxy defined in

Liberty Alliance and SAML version 2. This proxy has capabilities beyond that of a browser and can serve as a proxy to small mobile devices, such as cell phones. Enhancing this proxy to negotiate on behalf of the user removes the AAs ability to track where a user browses. The client could obtain the assertions needed for negotiation on demand from the AA but the proxy would negotiate for the user. Unlike an AA in BBTN, this proxy belongs to one user only. This would protect the privacy of the user and still yield some of the advantages of BBTN. Exploration of such a proxy extends beyond the scope of this thesis.

5.5.3 Single Point of Attack

Maintaining an on-line credential authority also represents a single point for an attack to obtain information about many users. A compromised authority could mean the release of all user information stored at that authority. Pfitzmann and Waidner [17] explained this problem with current single sign-on proposals. To minimize the threat, they recommend partitioning a user's information across several authorities, or even allowing a user-specific authority, or wallet, that holds information about only one user.

5.6 Unintended Credential Usage

Certificates can be stolen, lent, or pooled to gain un-intended access to systems. Using BBTN (and SAML in general) helps to limit these problems, but does not completely solve them.

BBTN helps limit the theft of credentials. Because negotiations occur directly between the AA and SP, AAs never place the user's credentials on the user's machine. This reduces the chance that rogue software on a user's computer could steal the credentials.

However, rogue software could capture a password, gaining the ability to use someone's credentials via the AA. For every use of a stolen password, a thief must contact the AA, possibly leaving an audit trail. This would help make detection of the theft easier.

Similar to credential theft, users cannot lend SAML bearer assertions. But again, nothing prevents lending passwords. More secure authentication mechanisms can help thwart this, such as biometrics, smart cards, proximity cards, etc. SAML does not specify which authentication mechanisms to use. The authentication mechanisms depend on the system deployed at the AA. Furthermore, SAML provides a way for an SP to request certain authentication mechanisms [9] to use when authenticating a user. A web server's release policies can directly reference the class of authentication methods used by the AA to authenticate a user.

Pooling credentials means that several users can combine their credentials to gain higher privileges. Using BBTN with different authentication mechanisms can help limit the pooling of credentials. In BBTN, pooling cannot occur within the same domain. Several users from the same AA cannot pool attribute assertions because of the tie between assertions and the user's current authenticated session. However, several users from different domains can still pool their credentials by having each user authenticate to different authorities using the same browser.

5.7 Denial of Service Attacks

BBTN requires AAs and SPs to keep track of several negotiation sessions. This can lead to denial of service (DOS) attacks.

Once the SP and AA begin negotiation, the SP may need to contact other authorities before it can complete the negotiation. Therefore, the negotiation session must remain open. An untrusted SP could flood an AA with uncompleted negotiation requests in an attempt to comsume all available memory on the AA.

However, before negotiation can begin, a user must authenticate to the AA. Therefore, a rogue SP would need to authenticate to the AA as a user in order to launch such an attack. To help prevent this, the AA should only allow one negotiation session for each user and SP. A successful attack would need either access to multiple accounts or access to one account and multiple rogue SP's.

Another scenario of a DOS attack involves a user attacking an SP. In this attack, the user attempts to access a protected web site and retrieve the VIPR page specifying the policy. The user would then redirect the SP to an AA with an authentication request. Then the user would abandon the connection, open a new connection and repeat the process. In this case the SP would have to remember which resource the user originally requested and how much of the policy the user had satisfied to that point. To minimize this threat, an SP must not store information for a user until that user has authenticated to at least one trusted AA.

5.8 Other Security Threats

SAML version 2.0 [8] includes a document, “Security and Privacy Considerations”, that explains countermeasures to several security and privacy threats:

- Eavesdropping

- Theft of the user authentication information or bearer token
- Replay
- Message insertion
- Message deletion
- Message modification
- Man-in-the-middle
- Impersonation without re-authentication

These same threats and countermeasures apply to BBTN. Countermeasures include a combination of "strong bilateral session authentication, confidentiality, and data integrity." These countermeasures are implemented with different combinations of the following mechanisms: XML encryption, XML signature, TLS, mutual authentication TLS, or HTTP basic authentication. For example, signing requests and responses, and sending them over TLS thwarts all the attacks listed above.

In BBTN, setting up these countermeasures can be difficult for an AA, since the SP is initially unknown. The AA can request a *metadata* document [6] containing the signing certificate of an SP. The AA uses the certificate to verify all signed requests and responses from the SP.

When an SP makes a request (such as an authentication request), it includes a provider ID. SAML recommends that the provider ID be a web page URL (Universal Resource Locater) that points to the metadata document. Usually, for SAML or Liberty, establishing secure communications with another provider involves importing the

metadata of the other provider. For BBTN, an AA must be able to dynamically discover the metadata for an SP, and therefore the provider ID must be the URL of a provider's metadata.

In BBTN, when an AA receives an authentication request, it must check to see if it has already obtained the metadata for the SP. If not, the AA must use the provider ID of the SP to obtain the metadata. This metadata should be requested using SSL/TLS to ensure integrety and authenticity. Once the AA obtains the metadata it applies the needed countermeasures to thwart the security and privacy attacks listed above.

Chapter 6 Contributions and Future Work

Browser-based trust negotiation avoids the problems associated with issuing certificates to users, simplifying negotiation for those who use it. BBTN allows people to use negotiating credential authorities with a web browser, by combining SAML single sign-on technology with trust negotiation.

This chapter explains the contributions of this thesis to the areas of single sign-on and trust negotiation, and discusses possible future work.

6.1 Contributions

Negotiating credential authorities and BBTN contribute several novel ideas to the area of single sign-on and trust negotiation.

6.1.1 Contributions to Single Sign-On

In Liberty Alliance and Shibboleth, an AA defines trust in an SP by identity. Instead, BBTN is an open system because trust in an SP is determined by attribute and not pre-configured by name. Similarly, BBAE does not base trust in SPs by identity. However, BBAE negotiates trust via privacy promises instead of credentials.

BBTN demonstrates how to extend existing single sign-on proposals (Liberty Alliance, Shibboleth, and SAML) to use trust negotiation as an authentication mechanism.

BBTN also adds the concept of combining credentials from multiple AAs to access a single SP to the area of single sign-on. This gives SPs additional assurance in a user's validity.

6.1.2 Contributions to Trust Negotiation

This thesis contributes several new ideas to the study of trust negotiation. First, this research contributes the concept of obtaining credentials on-demand from credential authorities.

Second, negotiating credential authorities provide an alternative approach for proxy based trust negotiation. Surrogate or mobile trust negotiation [18] uses proxies that represent a mobile user. Instead, a negotiating credential authority acts as a proxy for multiple users.

Third, using the SAML TN protocol, trust negotiation can utilize SAML assertions as digital credentials in addition to certificates.

Fourth, BBTN provides a browser-based, zero-footprint client to accomplish trust negotiation.

Finally, the VIPR service provides an interactive mechanism for a user to view and resolve unsatisfied policies.

6.2 Future Work

Future work should consider several variations and enhancements to both BBTN and SAML TN. One such variation, mentioned in section 5.5.2, could expand on the SAML 2 enhanced client proxy (ECP) profile. The profile adds additional functionality for clients that have enhanced capabilities beyond that of a browser. The client can also act as a proxy for the user when the user has access to only smaller or remote devices. Combining BBTN with ECP would create something similar to mobile trust negotiation with the

ability to obtain credentials as needed.

Another alternative to explore is a hybrid system supporting both traditional trust negotiation and BBTN. If users possess digital credentials, they can negotiate directly with the SP, and then resort to BBTN if they need additional credentials.

Bibliography

[1] R. Bradshaw, J. Holt, and K. E. Seamons. Concealing Complex Policies with Hidden Credentials. Eleventh ACM Conference on Computer and Communications Security, October 2004.

[2] M. Y. Becker and P. Sewell. Cassandra: Distributed Access Control Policies with Tunable Expressiveness. POLICY '04: Proceedings of the 5th International Workshop on Policies for Distributed Systems and Networks, pages 159–168, Yorktown Heights, NY, June 2004. IEEE Computer Society Press.

[3] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-X: A Peer-to-Peer Framework for Trust Establishment. IEEE Transactions on Knowledge and Data Engineering, 16(7):827–842, July 2004.

[4] S. Cantor and M. Erdos. Shibboleth-Architecture Draft v05. Internet 2 Middleware Working Group, May 2002. http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html

[5] S. Cantor, F. Hirsch, J. Kemp, R. Philpott, E. Maler. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/.

[6] S. Cantor, J. Moreh, R. Philpott, E. Maler. Metadata for the OASIS Security Assertion

Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/

[7] S. Cantor, J. Hughes, J. Hodges, F. Hirsh, P. Mishra, R. Philpott, E. Maler. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See http://www.oasis-open.org/committees/security/.

[8] F. Hirsch, R. Philpott, E. Maler. Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasisopen.org/committees/security/.

[9] J. Kemp, S. Cantor, P. Mishra, E. Maler. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS SSTC, March 2005. Document ID saml-authncontext-2.0-os. See http://www.oasis-open.org/committees/security/.

[10] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a Role-based Trust-management Framework. In 8th ACM Symposium on Access Control Models and Technologies, Como, Italy, June 2003

[11] Liberty Alliance. Liberty Alliance Project Phase 2 Specifications, November 2003. http://www.projectliberty.org/specs/

[12] W. Nejdl, D. Olmedilla, and M. Winslett. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. SDM '04: Workshop on Secure Data Management in a

Connected World, pages 118–132, Toronto, Canada, August 2004.

[13] OASIS eXtensible Access Control Markup Language Technical Committee. eXtensible Access Control Markup Language (XACML) Version 1.1 Specification. OASIS Standard, July 2003. http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf

[14] OASIS Security Services Technical Committee. Security Assertion Markup Language version 1.1 Specification. OASIS Standard, May 2003. http://www.oasis−open.org/committees/tc_home.php?wg_abbrev=security#samlv11

[15] OASIS Security Services Technical committee. Security Assertion Markup Language version 2.0 Specification. OASIS Standard, March 2005. http://www.oasis−open.org/committees/tc_home.php?wg_abbrev=security#samlv20

[16] Birgit Pfitzmann and Michael Waidner. BBAE -- a general protocol for browser-based attribute exchange. Research Report RZ 3455 (# 93800), IBM Research, September 2002.

[17] Birgit Pfitzmann and Michael Waidner. Privacy in browser-based attribute exchange. ACM Workshop on Privacy in the Electronic Society (WPES), Washington, Nov. 2002, pages 52-62, ACM Press

[18] T. van der Horst, T. Sundelin, K. E. Seamons, and C. Knutson. Mobile Trust Negotiation: Authentication and Authorization in Dynamic Mobile Networks. Eighth IFIP

Conference on Communications and Multimedia Security, Lake Windermere, England, September 2004.

[19] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated Trust Negotiation. DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.

Appendix XML Schema for SAML Trust Negotiation

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:byu:cs:irsl:negotiator:samltn"
    xmlns:samltn="urn:byu:cs:irsl:negotiator:samltn"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy">
    <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
    <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
    <import namespace="urn:oasis:names:tc:xacml:1.0:policy"
schemaLocation="cs-xacml-schema-policy-01.xsd"/>
    <annotation>
        <documentation>
            Document identifier: saml2-trustnegotiator.xsd
        </documentation>
      </annotation>
    <element name="NegotiationMessage" type="samltn:NegotiationMessageType"/>
      <complexType name="NegotiationMessageType">
         <complexContent>
            <extension base="samlp:RequestAbstractType">
               <choice>
                  <sequence>
                     <element ref="saml:Subject" minOccurs="0" />
                  </sequence>
                  <element ref="samlp:Response"/>
                  <!-- The Response possibly contains attribute statements in
assertions or a policy in the status detail -->
               </choice>
            </extension>
         </complexContent>
      </complexType>
    </element>

    <!-- A Response to this message results in a samlp:Response with the
status detail containing policy elements -->
</schema>
```